

CLAIMS

What is claimed is:

1. A method of reducing access latency to a task priority register of a local programmable interrupt controller unit within a microprocessor, the
5 method comprising:
receiving a command to write an interrupt mask value to the task priority register;
writing the interrupt mask value to the task priority register; and
writing the interrupt mask value into a shadow copy of the task priority
10 register,
wherein the shadow copy is written each time that the task priority register is written.
2. The method of claim 1, further comprising:
15 receiving a command to read the interrupt mask value from the task priority register;
reading the interrupt mask value from the shadow copy, instead of from the task priority register.
- 20 3. The method of claim 1, wherein the shadow copy is always written after the task priority register is written.
4. The method of claim 3, further comprising, upon receiving an interrupt,
25 reading the interrupt mask value from the task priority register and writing the interrupt mask value to the shadow copy.
5. The method of claim 1, wherein, if the task priority register is accessed frequently, then the shadow copy is stored in low-latency cache memory within the microprocessor.
30
6. The method of claim 1, wherein the method obviates a need to use a serialize instruction after the task priority register is written because each

interrupt performs a serialize operation and performs a read of an interrupt vector register.

7. The method of claim 6, whereby a latency of writing to the task priority register is substantially reduced.
8. The method of claim 2, whereby a latency of reading from the task priority register is substantially reduced.
9. The method of claim 1, wherein the microprocessor comprises an IPF architecture microprocessor.
10. The method of claim 9, wherein the local PIC unit in the microprocessor comprises a streamlined advanced programmable interrupt controller (SAPIC) unit in the IPF architecture microprocessor.
11. The method of claim 1, wherein the method is performed by an operating system.
12. The method of claim 9, wherein the method is performed by a UNIX-type operating system configured for the IPF architecture microprocessor.
13. The method of claim 9, wherein the method is performed by a version of a Windows operating system configured for the IPF architecture microprocessor.
14. An operating system with reduced access latency to a task priority register of a local programmable interrupt controller unit within a microprocessor, the operating system comprising:
microprocessor-executable code configured to write a priority level to the task priority register; and
microprocessor-executable code configured to write the priority level into a shadow copy of the task priority register,

wherein the shadow copy is written each time that the task priority register is written.

15. The operating system of claim 14, further comprising:
5 microprocessor-executable code configured to read the priority level from the shadow copy, instead of from the task priority register.
16. The operating system of claim 14, wherein the shadow copy is always written after the task priority register is written.
10
17. The operating system of claim 16, further comprising, microprocessor-executable code configured to, upon receipt of an interrupt, read the priority level from the task priority register and write the priority level to the shadow copy.
15
18. The operating system of claim 14, wherein, if the task priority register is accessed frequently, then the shadow copy is stored in low-latency cache memory within the microprocessor.
- 20 19. The operating system of claim 14, wherein the operating system avoids a need to use a serialize instruction after the task priority register is written because each interrupt performs a serialize operation.
20. The operating system of claim 19, whereby a latency of writing to the task
25 priority register is substantially reduced.
21. The operating system of claim 15, whereby a latency of reading from the task priority register is substantially reduced.
- 30 22. The operating system of claim 14, wherein the microprocessor comprises an IPF architecture microprocessor.

23. The operating system of claim 22, wherein the local PIC unit in the microprocessor comprises a streamlined advanced programmable interrupt controller (SAPIC) unit in the IPF architecture microprocessor.
- 5 24. The operating system of claim 22, wherein the operating system comprises a UNIX-type operating system configured for the IPF architecture microprocessor.
- 10 25. The operating system of claim 22, wherein the operating system comprises a version of a Windows operating system configured for the IPF architecture microprocessor.
- 15 26. A method of reducing a latency to read a task priority register of an IPF type microprocessor, the method comprising:
receiving a command to read an interrupt mask value from the task priority register; and
reading the interrupt mask value from the shadow copy at a memory location, instead of from the task priority register itself.
- 20 27. An operating system with reduced latency to read a task priority register of a local programmable interrupt controller unit within a microprocessor, the operating system comprising microprocessor-executable code configured read the interrupt mask value from the shadow copy at a memory location, instead of from the task priority register itself.
- 25 28. A multiple-processor computer system comprising:
a plurality of microprocessors interconnected by a processor bus, wherein each microprocessor includes a task priority register (TPR) with a interrupt mask value for that microprocessor;
30 a memory system, including local cache memory on each microprocessor and a main memory,
wherein the memory system holds data including an operating system and shadow copies of the TPRs, and

wherein the operating system includes executable-code for reading the interrupt mask values from the shadow copies and for maintaining the shadow copies.

- 5 29. A method of reducing latency to write a task priority register within a microprocessor, the method comprising:
upon receiving a command to write an interrupt mask value to the task priority register, writing the interrupt mask value to the task priority register without performing a serialization directly thereafter; and
10 upon receiving an interrupt, performing the serialization and reading an interrupt vector register, wherein a spurious indicator is returned if the interrupt is maskable.